Plus courts et plus longs chemins

Complément au chapitre 8 « Une voiture nous attend »

Soit I={1,2,…,n} un ensemble de tâches à ordonnancer. La durée d'exécution de chaque tâche i est connue et égale à p_i. Par hypothèse, une tâche en cours d'exécution ne peut pas être interrompue. L'exécution des tâches est soumise à des contraintes que certaines tâches ne peuvent pas commencer avant que d'autres soient terminées ou aient atteint un certain degré d'avancement. D'autres contraintes imposent que certaines tâches ne doivent pas démarrer trop tard après le début d'autres tâches. C'est le cas, par exemple, dans le cadre de processus chimiques où une trop longue attente entre deux tâches peut annuler la réaction chimique souhaitée. Notons t_i la période à laquelle la tâche i débute. Toutes ces contraintes peuvent être mises sous la forme suivante :

$$t_j - t_i \geq a_{ij}$$

où a_{ij} est un nombre réel donné correspondant à une durée. Voici quelques exemples :

- la tâche j ne peut pas commencer avant que la tâche i soit terminée : $t_j - t_i \geq p_i$
- la tâche j commence au plus tard lorsqu'un certain laps de temps T s'est écoulé depuis le début de la tâche i : $t_i + T \geq t_j$ ce qui est équivalent à $t_i - t_j \geq -T$
- les tâches i et j doivent débuter simultanément : $t_i - t_j \geq 0$ et $t_j - t_i \geq 0$
- la tâche j doit succéder immédiatement à la tâche i, sans qu'il y ait interruption : $t_i - t_j \geq -p_i$ et $t_j - t_i \geq p_i$

Le problème auquel on s'intéresse ici consiste à trouver un ordonnancement de durée minimale. Il s'agit en d'autres termes d'ordonnancer les tâches de telle sorte que l'heure à laquelle l'exécution de la dernière tâche prend fin soit aussi petite que possible.

Formulation en termes de graphes

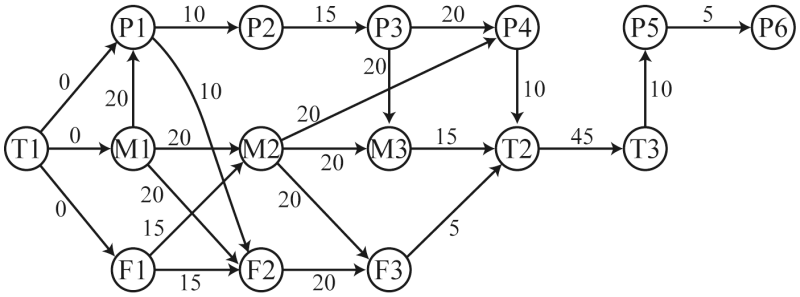
On peut associer un graphe à ce problème. Pour cela, on considère deux tâches fictives 0 et n+1.

- À chaque tâche $i \in \{0,1,2,\ldots,n,n+1\}$ correspond un sommet du graphe.
- Les sommets i et j liés par la relation $t_j - t_i \geq a_{ij}$ sont reliés par un arc de longueur a_{ij} allant de i vers j.
- Chaque sommet $i \in \{1,2,\ldots,n\}$ est relié au sommet n+1 par un arc de longueur p_i.
- Le sommet 0 est relié à chaque sommet $i \in \{1,2,\ldots,n\}$ par un arc de longueur nulle.

Trouver un ordonnancement de durée minimale revient alors à déterminer le plus long chemin entre les sommets 0 et n+1. Un tel chemin est appelé *chemin critique* car les tâches sur ce chemin ont leur période de début imposée si l'on désire terminer l'ensemble des tâches au plus tôt. Les tâches situées sur un chemin critique sont appelées *tâches critiques*. La détermination des chemins critiques permet de focaliser l'attention sur l'exécution des tâches critiques : tout retard pris sur l'exécution d'une tâche critique allonge inévitablement la durée du projet.

Pour tenter de retarder le réveil de la famille Courtel, Manori construit un tel graphe. À la liste des tâches décrites par Sébastien, on peut construire le graphe ci-dessous.

Tâches	Descriptions	Durées	Prédécesseurs
P1	Le père fait sa toilette	10	T1, M1
P2	Vider les armoires dans la chambre des parents	15	P1
P3	Faire les valises des parents	20	P2
P4	Apporter les valises à la réception	10	P3, M2
P5	Rendre les clés	5	T3
P6	Prendre possession de la voiture	0	P5
M1	La mère fait sa toilette	20	T1
M2	Faire les valises des filles	20	M1, F1
M3	Se maquiller et vérifier la chambre des parents	15	P3, M2
F1	Vider les armoires dans la chambre des filles	15	T1
F2	Les filles font leur toilette	20	F1, P1, M1
F3	Vérifier la chambre des filles	5	F2, M2
T1	Se lever	0	
T2	Prendre le petit-déjeuner	45	P4, M3, F3
T3	Dernier brossage des dents	10	T2



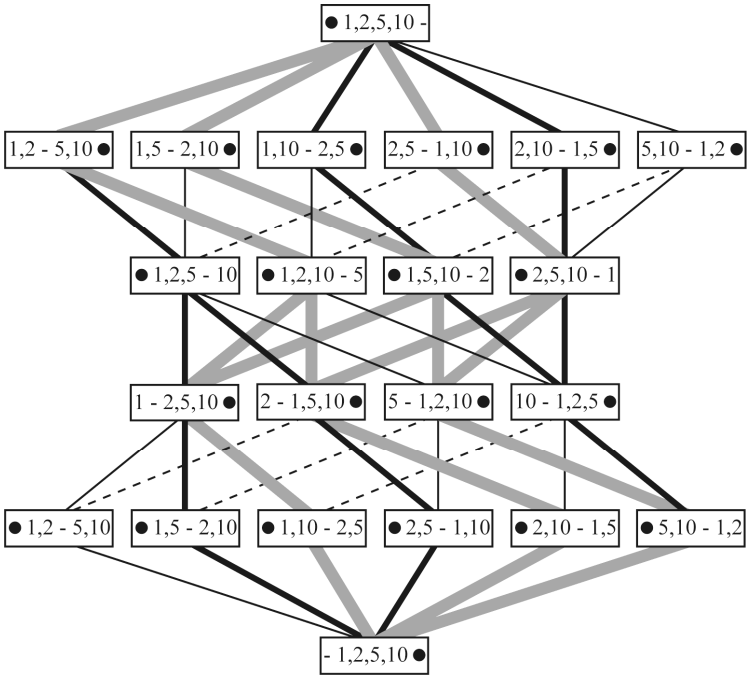
Ici, la tâche 0 est en fait la tâche T1 et la tâche n+1 est la tâche P6. On a par exemple la contrainte que la tâche M1 précède P1, ce qui signifie que $t_{P1} \geq t_{M1}+20$. On peut réécrire cette inégalité sous forme standard pour obtenir $t_{P1}-t_{M1} \geq 20$, ce qui explique l'arc de longueur 20 allant de M1 vers P1.

Dans ce graphe, T1 n'est relié directement qu'à P1, M1 et F1. Théoriquement, tel que soulevé par Sébastien, on aurait aussi dû mettre un arc de T1 vers les autres sommets. Mais ces arcs sont en fait inutiles car tout chemin allant de T1 vers ces autres sommets doit passer par P1, M1 ou F1.

Le plus long chemin de T1 à P6 dans ce graphe est de longueur 140, ce qui signifie que la famille Courtel a besoin de 2 heures et 20 minutes pour se préparer le matin du départ.

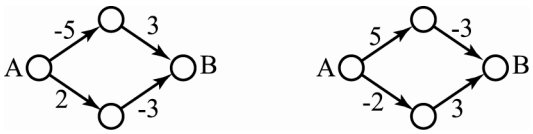
Alors que le problème que nous venons d'étudier est la recherche d'un plus long chemin dans un graphe, d'autres problèmes consistent à déterminer un plus court chemin. Par exemple, lorsqu'on veut se rendre d'un point à un autre d'un réseau routier, on peut rechercher le chemin le plus court ou le chemin le plus rapide. C'est ce que font tous les systèmes GPS. Lorsqu'on veut minimiser la distance parcourue, les valeurs sur les arcs sont les distances à parcourir d'un point à un autre. Si l'on veut par contre que le trajet soit aussi rapide que possible, les valeurs mises sur les arcs sont les durées des trajets.

Aussi, lorsqu'on recherche la meilleure stratégie pour passer d'un état A à un état B, on peut considérer les étapes intermédiaires par lesquelles on peut passer. On peut aussi mesurer le coût à payer pour se rendre d'une étape à une autre. La meilleure stratégie consistera alors à emprunter le plus court chemin de A à B. C'est exactement ce qu'a fait Manori pour résoudre le problème de l'évasion des 4 prisonniers. Il a dessiné le graphe suivant.



Le problème consiste à se rendre du sommet au haut du graphe qui représente la situation où tous les prisonniers sont dans leur cellule, au sommet au bas du graphe qui correspond à la situation où tous les prisonniers sont libres. Le plus court chemin donne la stratégie d'évasion la plus courte en temps.

Nous allons maintenant indiquer comment déterminer un plus court chemin dans un graphe. Notons que si c'est un plus long chemin qu'il faut déterminer, il suffit de changer de signe chaque longueur sur les arêtes (c'est-à-dire remplacer chaque longueur d_{ij} par $-d_{ij}$) et de rechercher un plus court chemin. Par exemple, le plus long chemin de A à B dans le graphe à gauche ci-dessous est celui du bas de longueur -1, et le plus court dans le graphe à droite est aussi celui du bas avec une longueur 1.



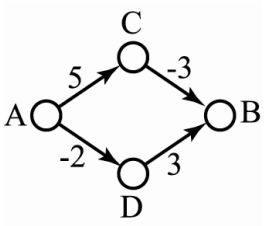
Certains pourraient être surpris qu'on considère des longueurs négatives car dans un réseau routier, toutes les distances sont positives et les durées des trajets sont positives aussi. Mais nous avons vu que la recherche de plus courts ou plus longs chemins peut apparaître dans d'autres contextes. Par exemple, si on demande à ce que la tâche A commence au plus tard 20 minutes après le début de la tâche B, on a la contrainte $t_A \leq t_B+20$, ce qui s'écrit aussi $t_B - t_A \geq -20$. Dans le graphe considéré nous mettons donc un arc de A vers B de longueur négative -20.

Nous commençons par présenter un algorithme de recherche de plus court chemin entre un sommet A et tous les autres sommets du graphe dans le cas où le graphe considéré n'a pas de circuit. Cet algorithme marque un sommet dès que la longueur du plus court chemin jusqu'à lui est connue. Notons PCC(v) la longueur du plus court chemin de A au sommet v et d_{xy} la longueur de l'arc allant de x vers y.

Algorithme de recherche d'un plus court chemin entre un sommet A et tous les autres sommets du graphe dans le cas où le graphe considéré n'a pas de circuit

1. poser PCC(A)=0 et marquer A ;
2. s'il existe au moins un sommet non marqué alors aller à 3, sinon STOP.
3. choisir un sommet v non marqué dont tous les prédécesseurs sont marqués. Déterminer le minimum des valeurs $PCC(w)+d_{wv}$ en considérant tous les prédécesseurs w de v et poser PCC(v) égal à ce minimum ; marquer le sommet v et retourner à 2.

Pour le petit exemple ci-dessous, après avoir posé PCC(A)=0 et avoir marqué A, on peut choisir v égal à C ou D. Choisissons v=C. On pose donc PCC(C)=PCC(A)+5=5 et on marque C. Le seul sommet dont tous les prédécesseurs sont marqués est alors le sommet D. On pose donc PCC(D)=PCC(A)-2=-2 et on marque D. Finalement le dernier sommet à considérer est le sommet B. On pose PCC(B)=min{PCC(C)-3, PCC(D)+3}=min{2,1}=1. L'algorithme s'arrête après avoir marqué B.



Lorsque le graphe considéré a des circuits, l'algorithme ci-dessus ne peut plus être appliqué car il n'existe plus forcément de sommet dont tous les prédécesseurs sont marqués. En effet, avant de marquer le premier sommet d'un circuit, chaque sommet de ce circuit a un prédécesseur non marqué.

Le deuxième algorithme que nous présentons n'est valide que dans les cas où toutes les distances sont positives. Cet algorithme effectue également un marquage des sommets, un après l'autre. Il considère des valeurs temporaires pcc(v) pour le plus court chemin de

A à v. Ces valeurs temporaires sont mises à jour pour finalement aboutir à la vraie valeur PCC(v) une fois que le sommet v est marqué. L'algorithme peut être décrit comme suit.

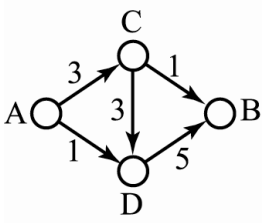
Algorithme de recherche d'un plus court chemin entre un sommet A et tous les autres sommets du graphe dans le cas où toutes les distances sont positives

1. poser pcc(A)=0 et pcc(v) = ∞ (infini) pour tout v≠A ;
2. s'il existe au moins un sommet non marqué alors aller à 3, sinon STOP.
3. choisir un sommet v non marqué de valeur pcc(v) minimale, poser PCC(v)=pcc(v) et marquer v ;
4. poser pcc(w)=min{pcc(w), PCC(v)+d_{vw}} pour tout successeur w de v non marqué (avec v égal au dernier sommet marqué à l'étape 3) et retourner à 2.

Pour l'exemple ci-dessous, après avoir posé pcc(A)=0 et pcc(B)=pcc(C)=pcc(D)=∞, on choisir v=A et on pose PCC(A)=0 tout en marquant A. On fait alors les mises à jour suivantes :

- pcc(C)=min{∞, PCC(A)+3}=3
- pcc(D)=min{∞, PCC(A)+1}=1

On choisit donc v=D comme prochain sommet et on pose PCC(D)=1 en marquant D. La seule nouvelle mise à jour est pcc(B)=min{∞, PCC(D)+5}=6.
On choisit ensuite v=C et on pose PCC(C)=3 en marquant C et on fait la mise à jour pcc(B)=min{6, PCC(C)+1}=4.
Finalement, on choisit v=B, on pose PCC(B)=4 en marquant B et l'algorithme s'arrête.



Remarquons que cet algorithme ne produit pas forcément un plus court chemin de A aux autres sommets si le graphe contient quelques distances négatives. Par exemple, si on modifie la distance sur l'arc allant de C à D pour la rendre égale à -3 au lieu de 3, l'algorithme proposé débutera comme indiqué ci-dessus et posera donc PCC(D)=1 alors que le plus court chemin de A à D est de longueur 0 (en passant pas C).

Nous indiquons finalement comment déterminer un plus court chemin dans un graphe pouvant contenir des circuits et des arcs avec des distances négatives. Nous supposons qu'il n'existe pas de circuit de longueur négative car, sinon, le problème n'aurait pas de solution de valeur finie. Il suffirait en effet de se rendre de A vers ce circuit et de tourner indéfiniment dans le circuit avant de ressortir pour se rendre vers un autre sommet pour avoir un chemin de longueur -∞. L'algorithme qui suit ne considère que des valeurs PCC(v) pour chaque sommet v ; ces valeurs sont mises à jour tout au cours de l'algorithme et nous ne savons qu'elles correspondent aux longueurs des plus courts chemins de A aux autres sommets qu'à la toute fin de l'algorithme.

Algorithme de recherche d'un plus court chemin entre un sommet A et tous les autres sommets du graphe dans le cas général (mais en supposant qu'il n'existe pas de circuit de longueur négative)

1. poser PCC(A)=0 et PCC(v) = ∞ (infini) pour tout v≠A ;
2. pour chaque sommet v, calculer le minimum des valeurs PCC(w)+d_{wv} en considérant tous les prédécesseurs w de v et poser f(v) égal à ce minimum
3. pour chaque sommet v, faire la mise à jour PCC(v)=min{PCC(v), f(v)}
4. Si la valeur PCC(v) a été modifiée à l'étape 3 pour au moins 1 sommet v alors retourner à 2, sinon STOP.

Pour l'exemple ci-dessous, on pose PCC(A)=0, PCC(B)=PCC(C)=PCC(D)=∞ et on calcule

- f(B)= PCC(D)+5 = ∞+5 = ∞
- f(C)= min{PCC(A)+3, PCC(B)+1 = min{0+3, ∞+1}= 3
- f(D)= min{PCC(A)+1, PCC(C)-3}= min{0+1, ∞-3}= 1

On doit donc mettre à jour PCC(C) qui devient égal à 3 et PCC(D) qui devient égal à 1.
Comme deux valeurs ont été modifiées à l'étape 3, on recalcule les valeurs f(v) :

- f(B)= PCC(D)+5 = 1+5 = 6
- f(C)= min{PCC(A)+3, PCC(B)+1 = min{0+3, ∞+1}= 3
- f(D)= min{PCC(A)+1, PCC(C)-3}= min{0+1, 3-3}= 0

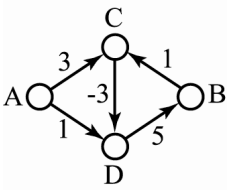
On doit donc mettre à jour PCC(B) qui devient égal à 6 et PCC(D) qui devient égal à 0.
Comme deux valeurs ont été modifiées à l'étape 3, on recalcule les valeurs f(v) :

- f(B)= PCC(D)+5 = 0+5 = 5
- f(C)= min{PCC(A)+3, PCC(B)+1 = min{0+3, 6+1}= 3
- f(D)= min{PCC(A)+1, PCC(C)-3}= min{0+1, 3-3}= 0

On doit donc mettre à jour PCC(B) qui devient égal à 5.
Comme une valeur a été modifiée à l'étape 3, on recalcule les valeurs f(v) :

- f(B)= PCC(D)+5 = 0+5 = 5
- f(C)= min{PCC(A)+3, PCC(B)+1 = min{0+3, 5+1}= 3
- f(D)= min{PCC(A)+1, PCC(C)-3}= min{0+1, 3-3}= 0

Comme aucune valeur f(v) n'est inférieure à PCC(v), l'algorithme s'arrête.



Dans les algorithmes que nous avons décrits, il est facile de retracer le plus court chemin de A vers chacun des autres sommets du graphe. Il suffit pour cela de mémoriser d'où vient l'arc qui entre en v lorsqu'on modifie PCC(v) (ou pcc(v) pour l'algorithme précédent). Ici, par exemple, la dernière mise à jour de PCC(B) vient de D, celle de PCC(C) vient de A et celle de PCC(D) vient de C. Les arcs utilisés pour les plus courts chemins sont donc les suivants.